Amendments to the Claims

The Claim Listing below replaces all prior versions of the claims in the subject application.

Claim Listing:

Claims 1 - 60 (cancelled).

61 (currently amended). Machine-readable memory storing instructions that when executed result in a processor performing operations comprising:

assigning received packets to threads for processing, the threads to be executed by microengines in the processor, the assigning being in accordance with thread scheduling information that includes thread capabilities information, port-to-thread assignments listing, and thread busy tracking information, the thread capabilities information indicating receive processing threads capabilities and appropriate port protocol servicing capabilities, the port-to-thread assignments listing maintaining a current list of active receive processing thread-to-port assignments, the thread busy tracking information being maintained by a thread busy mask register that indicates any threads that are actively servicing ports; and

in event that a processing exception occurs for a particular thread that is processing a particular received packet, sending the particular received packet to a processor core for further processing by the processor core;

the processor core and the microengines being coupled via an input/output (I/O) bus interface to ports to receive the received packets, the processor core and the microengines to issue commands to another interface comprised in the I/O bus interface to access shared resources of the I/O bus interface, the another interface to arbitrate which of the commands to service and to move data among the shared resources, the processor core, and the microengines, the shared resources including a hash unit to be accessed by the microengines during lookup operations and to generate hash indexes.

62 (Previously Presented). The machine-readable memory of claim 61, wherein:

all of the threads executed by the microengines are able to branch based on global signaling states; and

the global signaling states are used to determine one of:

resource availability; and

whether resource servicing is due.

63 (Previously Presented). The machine-readable memory of claim 61, wherein:

the threads executed by the microengines include a scheduler thread to maintain the thread capability information and the port-to-thread assignments.

64 (Previously Presented). The machine-readable memory of claim 63, wherein:

the thread capability information indicates to the scheduler thread capabilities of other threads and which of the other threads may be appropriate to service a certain particular port.

65 (Previously Presented). The machine-readable memory of claim 61, wherein:

the threads executed by the microengines include a processing thread to parse a header of the particular received packet and to perform a lookup based on the header;

unless the processing exception occurs, the processing thread is to store the particular received packet in memory and to enqueue the particular received packet by placing a packet link descriptor for the particular received packet in a transmit queue associated with a forwarding port indicated by the lookup; and

if the processing exception occurs, the processing thread is to perform the sending of the particular received packet.

66 (previously presented). The machine-readable memory of claim 65, wherein:

the threads executed by the microengines also include a transmit processing thread, an arbiter thread, and a transmit scheduler thread;

the transmit scheduler thread is to assign the particular received packet to the transmit processing thread;

the transmit processing thread is to send the particular received packet to the forwarding port; and

the arbiter thread is to prioritize transmission queues.

67 (currently amended). The machine-readable memory of claim 61, wherein:

~~the processor core and the microengines are coupled via an input/output (I/O) bus interface to ports to receive the received packets; and~~

the processor core to make function calls through an operating system to operate on the microengines.

68 (Previously Presented). The machine-readable memory of claim 61, wherein the respective microengine includes:

general purpose registers coupled to an arithmetic logic unit;

multiple program counters coupled to a control store to receive the microcode loaded into the respective microengine via the processor core; and

context switching logic coupled to the program counters.

69 (currently amended). A method comprising:

assigning received packets to threads for processing, the threads to be executed by microengines in a processor, the assigning being in accordance with thread scheduling information that includes thread capabilities information, port-to-thread assignments listing, and thread busy tracking information, the thread capabilities information indicating receive processing threads capabilities and appropriate port protocol servicing capabilities, the port-to-thread assignments listing maintaining a current list of active receive processing thread-to-port assignments, the thread busy tracking information being maintained by a thread busy mask register that indicates any threads that are actively servicing ports; and

in event that a processing exception occurs for a particular thread that is processing a particular received packet, sending the particular received packet to a processor core for further processing by the processor core;

the processor core and the microengines being coupled via an input/output (I/O) bus interface to ports to receive the received packets, the processor core and the microengines to issue commands to another interface comprised in the I/O bus interface to access shared resources of the I/O bus interface, the another interface to arbitrate which of the commands to service and to move data among the shared resources, the processor core, and the microengines, the shared resources including a hash unit to be accessed by the microengines during lookup operations and to generate hash indexes.

70 (Previously Presented). The method of claim 69, wherein:

all of the threads executed by the microengines are able to branch based on global signaling states; and

the global signaling states are used to determine one of:

resource availability; and

whether resource servicing is due.

71 (Previously Presented). The method of claim 69, wherein:

the threads executed by the microengines include a scheduler thread to maintain the thread capability information and the port-to-thread assignments.

72 (Previously Presented). The method of claim 71, wherein:

the thread capability information indicates to the scheduler thread capabilities of other threads and which of the other threads may be appropriate to service a particular port.

73 (Previously Presented). The method of claim 69, wherein:

the threads executed by the microengines include a processing thread to parse a header of the particular received packet and to perform a lookup based on the header;

unless the processing exception occurs, the processing thread is to store the particular received packet in memory and to enqueue the particular received packet by placing a packet link descriptor for the particular received packet in a transmit queue associated with a forwarding port indicated by the lookup; and

if the processing exception occurs, the processing thread is to perform the sending of the particular received packet.

74 (Previously Presented). The method of claim 73, wherein:

the threads executed by the microengines also include a transmit processing thread, an arbiter thread, and a transmit scheduler thread;

the transmit scheduler thread is to assign the particular received packet to the transmit processing thread;

the transmit processing thread is to send the particular received packet to the forwarding port; and

the arbiter thread is to prioritize transmission queues.

75 (currently amended). The method of claim 69, wherein:

~~the processor core and the microengines are coupled via an input/output (I/O) bus interface to ports to receive the received packets; and~~

the processor core to make function calls through an operating system to operate on the microengines.

76 (Previously Presented). The method of claim 69, wherein the respective microengine includes:

general purpose registers coupled to an arithmetic logic unit;

multiple program counters coupled to a control store to receive the microcode loaded into the respective microengine via the processor core; and

context switching logic coupled to the program counters.